Improving Cloud Gaming traffic QoS: a comparison between class-based queuing policy and L4S

Philippe Graff, Xavier Marchal, Thibault Cholez, Bertrand Mathieu, Stéphane Tuffin, Olivier Festor

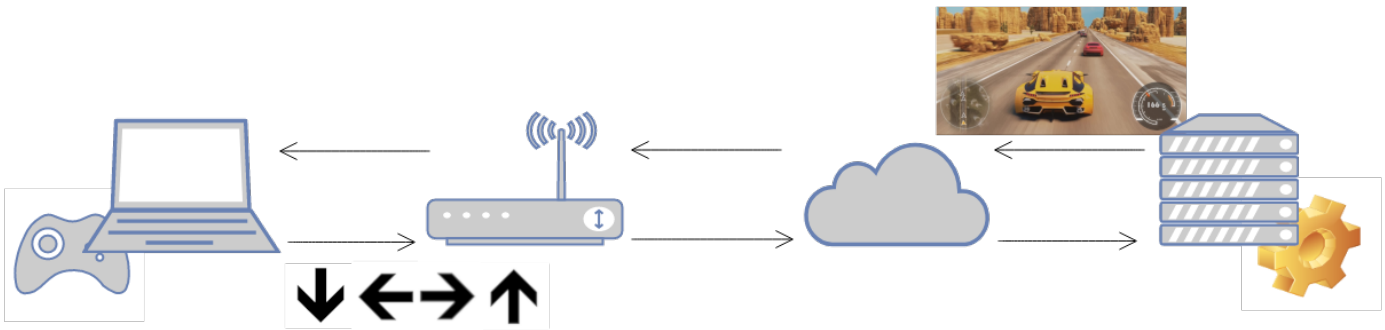
Université de Lorraine, CNRS, Inria, LORIA, Nancy, France
Orange Innovation, Lannion, France

17/03/25 - ICCRG@IETF 122

An example of low-latency service: Cloud Gaming (CG)

Principles

- Game is executed on a remote server $\neq$ Online games
- User interacts with a thin client sending inputs and displaying a multimedia flow
- Examples: Nvidia GeForce Now, Amazon Luna, Microsoft xCloud, Sony PlayStation Now, Google Stadia (RIP), etc.



An example of low-latency service: Cloud Gaming (CG)

Pros and Cons

- + Instant access to many games (no installation, no update)
- + No expensive hardware on client's side (noise, energy consumption)
- + Hardware mutualization on server's side
- - Need for a "good" network access
- - Carbon footprint can be worst depending on games/play time

CG adoption

- Steady grow over the past few years, dozens of millions of users
- Expected to become a significant part of Internet traffic share

Cloud Gaming traffic

CG traffic properties

- Relies on Real Time Protocol / RTCP over UDP (webRTC)
- High bandwidth: 1080p to 4K @60 fps, lower compression (real-time encoding)
- Short playout buffer, no retransmission (replaced by FEC)
- Proprietary Congestion Control Algorithms (CCA)

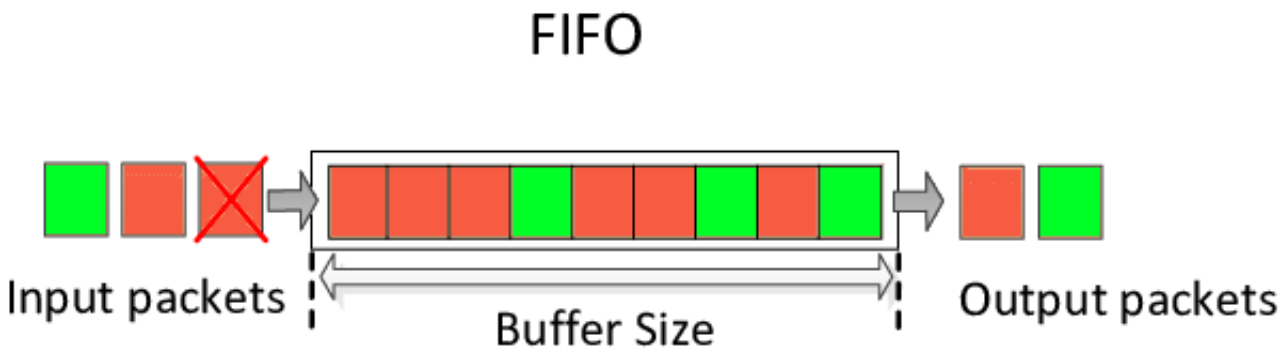
CG traffic QoS

- CG service is sensitive to latency: 100ms budget for the E2E latency chain "motion to photon"
- Network QoS requires **at the same time**:
  - High bandwidth (30Mb/s)
  - Low and stable latency (low jitter)
  - No loss

Bufferbloat phenomenon

Causes

- Network devices implement large buffers to maximize bandwidth
- Best effort flows and low-latency flows share the same buffer
- Loss based CCA fill the buffer at the bottleneck
- Traditional AQM apply a single policy irrespective of the differences in delay-throughput objectives of applications



Bufferbloat phenomenon

Consequences

- High queuing latency in case of congestion
- Poor cohabitation between loss-based and latency-based CCA

Solution

- Isolate different types of traffic: loss-based and latency-sensitive
- Obvious way: use separate queues

Queue Management solutions

Hierarchical Token Bucket (HTB) queuing discipline

- Considers traffic classes. Previous work can identify CG flows at line rate
- Dedicated short queue for CG
- Prioritization of CG traffic

Low Latency, Low Loss, and Scalable Throughput (L4S)

- Considers 2 traffic classes: classic Best Effort (BE) and scalable Low-Latency (LL)
- Relies on IP header Explicit Congestion Notification (ECN) field
- Application's CCA must comply with L4S requirements

Problem statement

Main research questions

- Is Cloud Gaming traffic affected by bufferbloat?
- Can dual-queue solutions improve Cloud Gaming traffic QoS and by how much?

Introduction

Motivation

- Previous open-source platform "Gaming Anywhere" uses TCP
- Need for a more representative and L4S-compatible CG platform
- Link: `https://github.com/mosaico-anr/eCGP`

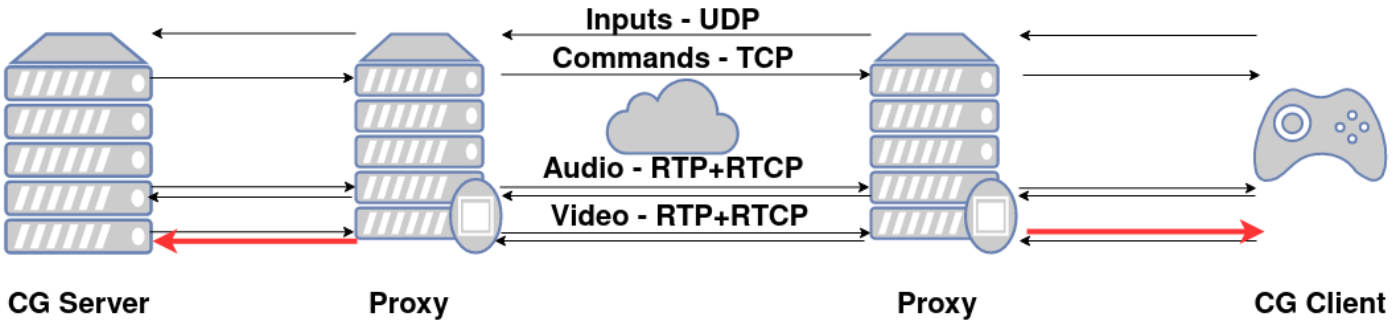
Features

- Relies on X11/ALSA (linux only) for recording and on FFmpeg API to process audio and video streams
- Supported codecs H.264 and 265 (video), Opus (audio), GPU encoding with NVENC
- More details on the software architecture / implementation in the paper

SCReAM integration

SCReAM as a proxy

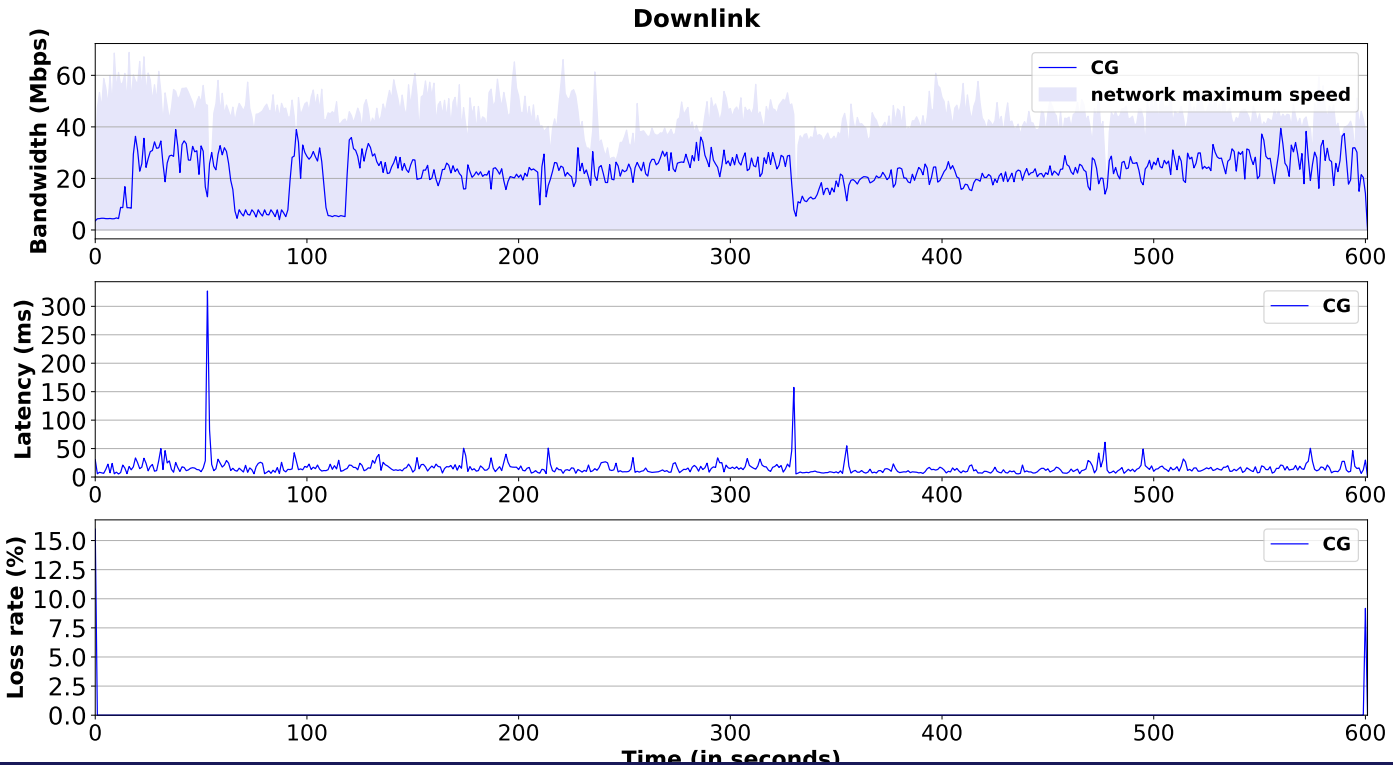
- SCReAM is an open-source CCA developed by Ericsson for RTC over RTP/RTCP
- Only L4S-compatible CCA for RTP
- Integrated as a proxy: modular architecture to easily change the CCA
- Drives the RTP video flow: regularly sends **Bitrate** and **I_Frame** requests to the encoder



Assessment

Evaluation under 4G cellular network conditions

- CG Client and CG server are separated by a router
- Cellular network emulation with Mahimahi: replays traces of transmissions opportunities (txops) recorded on a real network

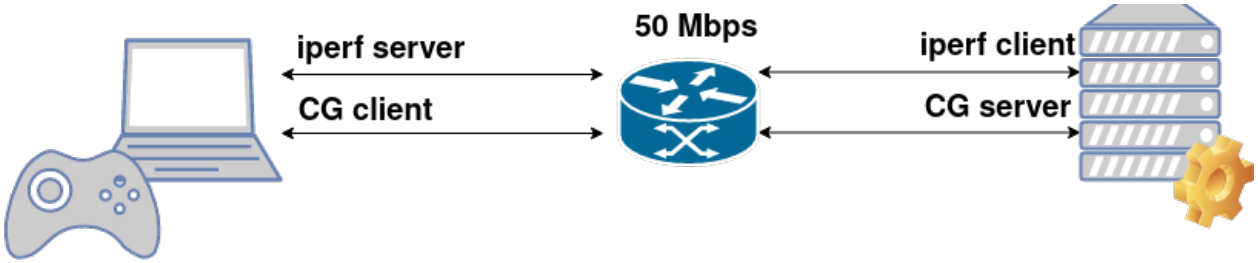


Assessment

Table 1: Queuing delays measured with the same cellular network trace and the same game for different CCAs

	No CCA	SCReAM CCA	GFN CCA
Average Latency	47.9ms	15.98ms	-
% of packets $>$ 10ms	99.8%	73.9%	52.4%
% of packets $>$ 20ms	85.2%	18.4%	30.4%
% of packets $>$ 50ms	21.1%	1.5%	10.0%
% of packets $>$ 100ms	6.8%	0.3%	2.6%

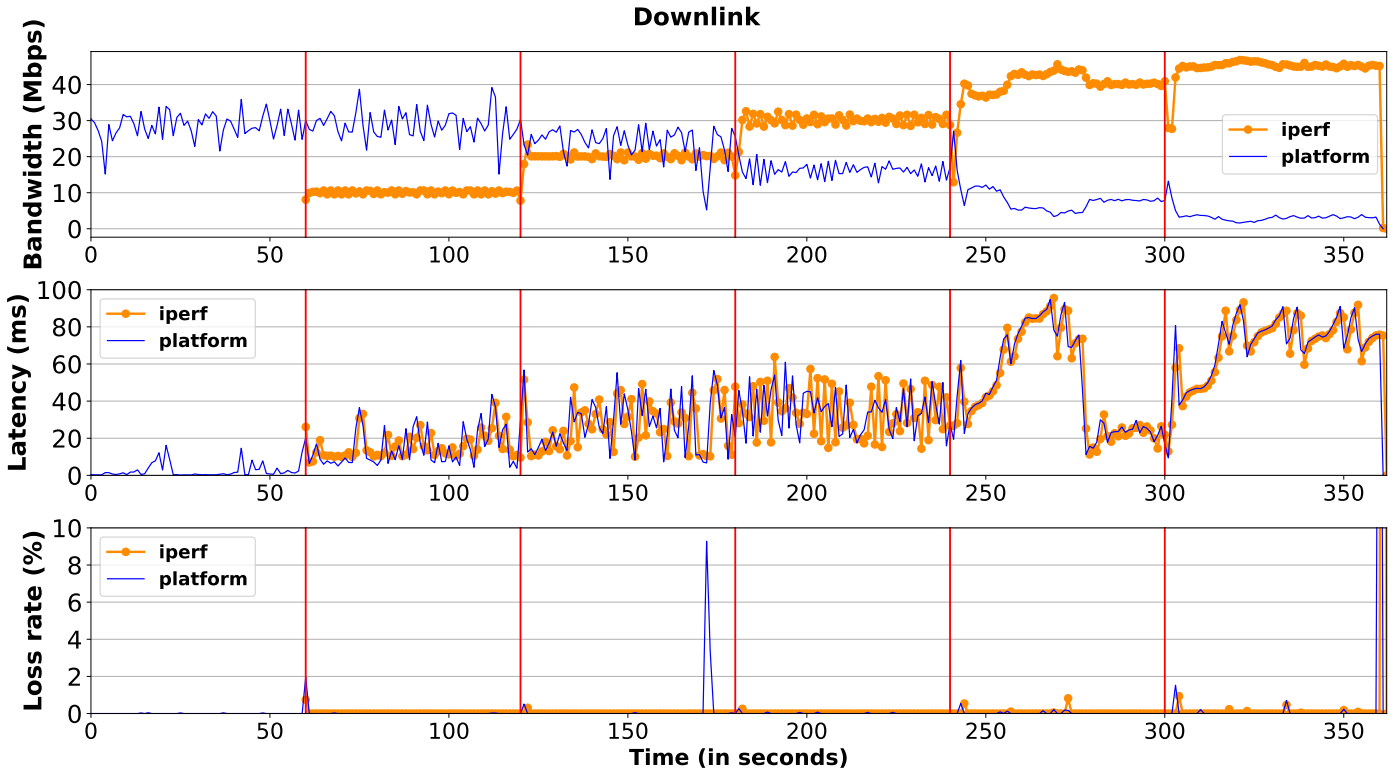
Testbed



Experimental protocol

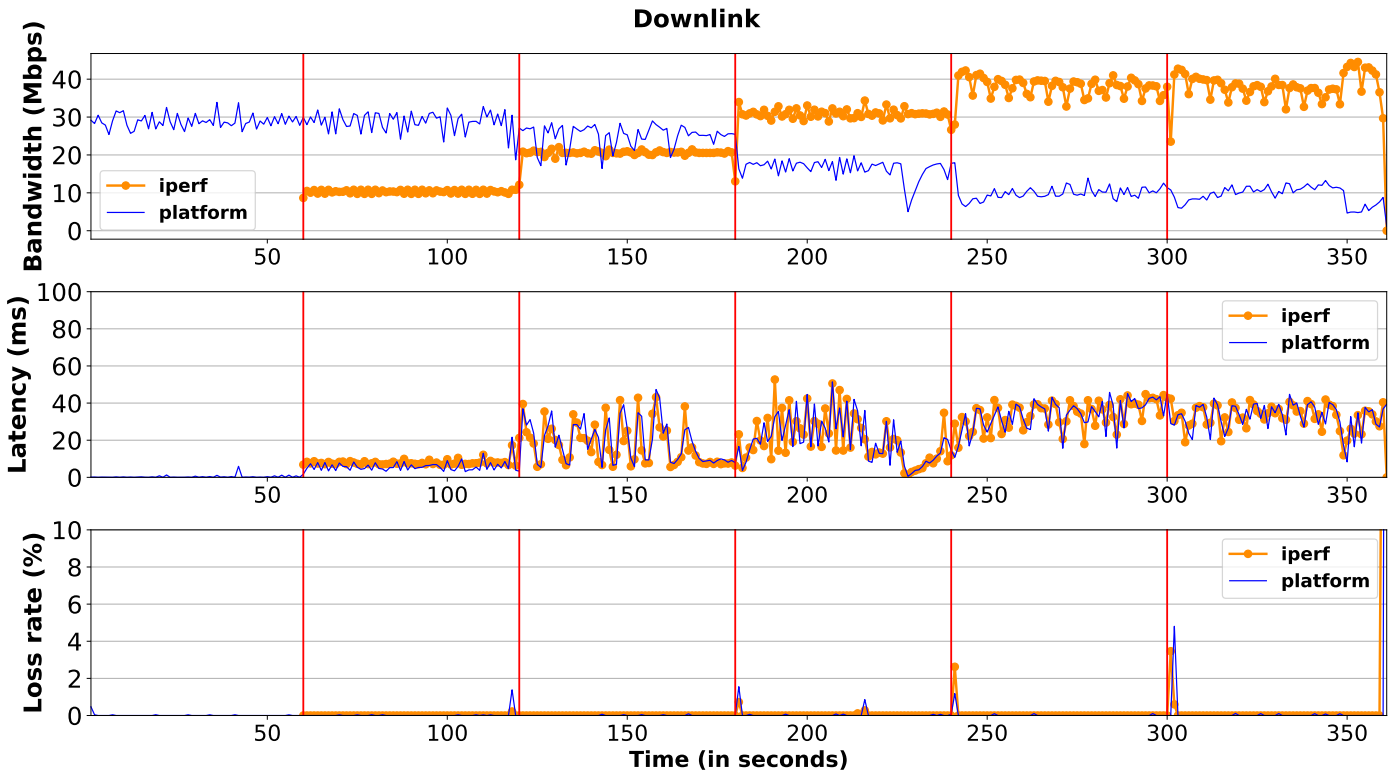
- Bottleneck router has a 50mb/s bandwidth and emulate a 30ms RTT
- Implements in turn different queuing solutions: Droptail, HTB, L4S
- Competing flows to CG made by iperf with different CCA (TCP CUBIC and TCP BBRv2) and different target bitrates (0, 10, 20, 30, 45 Mb/s)
- Throughput, queuing latency and loss rate are monitored

Droptail: SCReAM vs Cubic



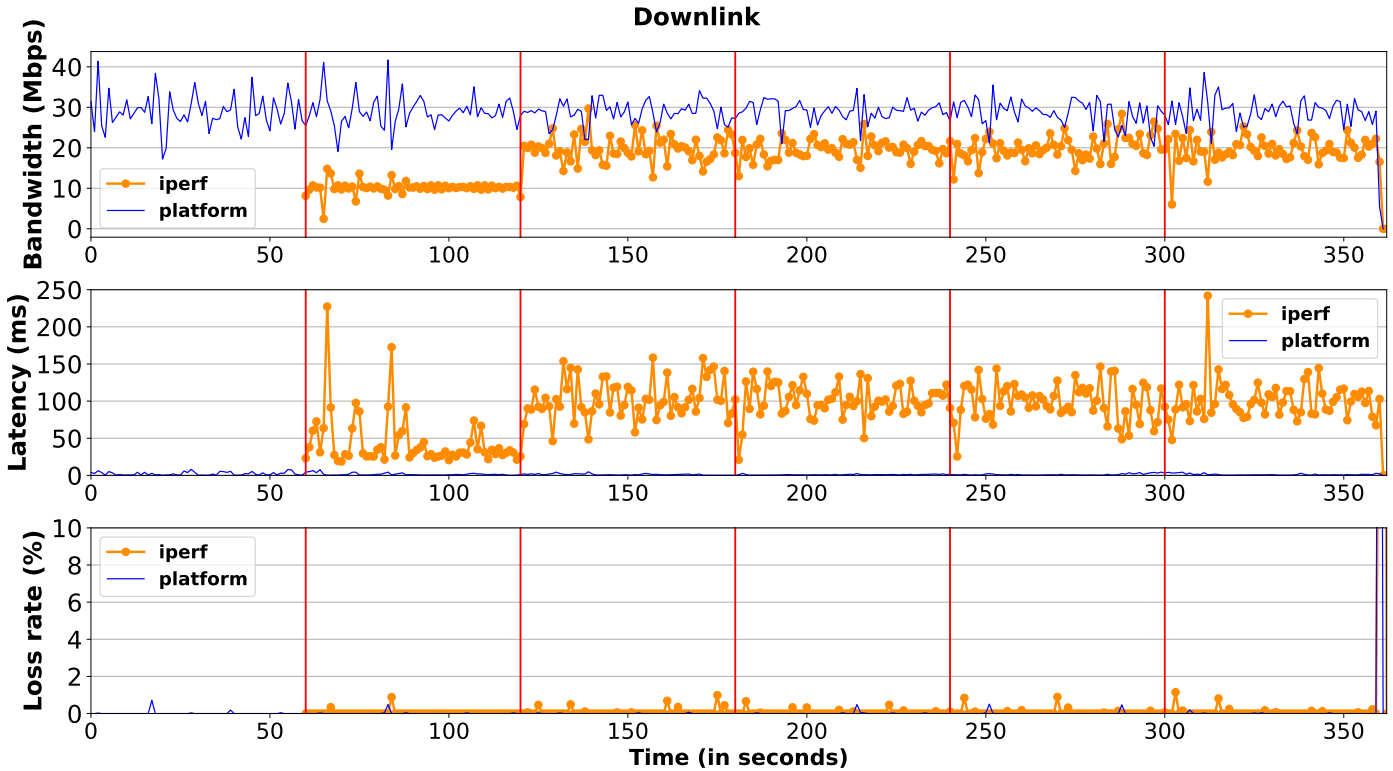
- Bufferbloat is real, same high queuing latency for both flows (avg 45ms), and fluctuating (24ms of jitter)
- Unfair cohabitation: without loss CUBIC steals the bandwidth (avg 12Mb/s for CG vs 33Mb/s for iperf)

Droptail: SCReAM vs BBRv2



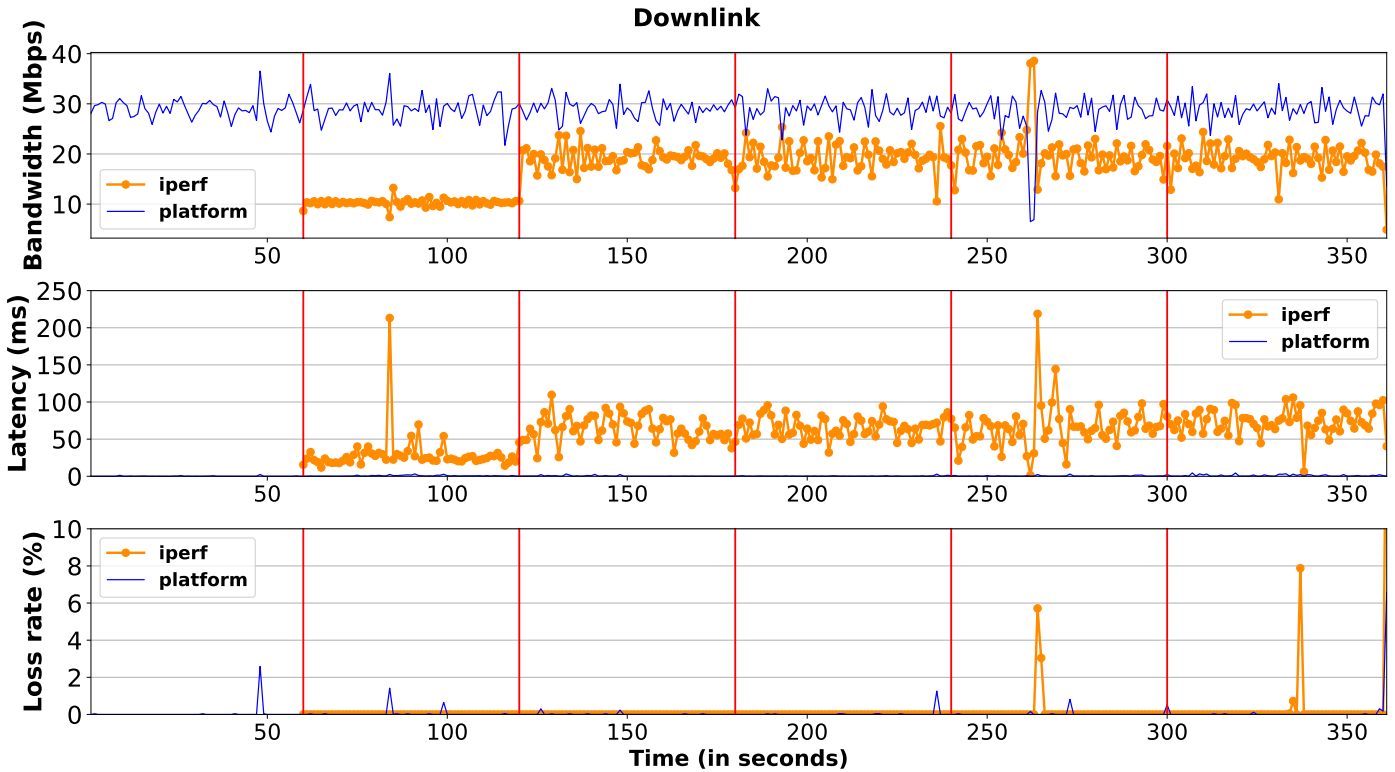
- Better than CUBIC regarding queuing latency (avg 26ms), more stable
- Bandwidth sharing still very unfair (avg 15Mb/s for CG vs 31Mb/s for iperf)

HTB: SCReAM vs Cubic



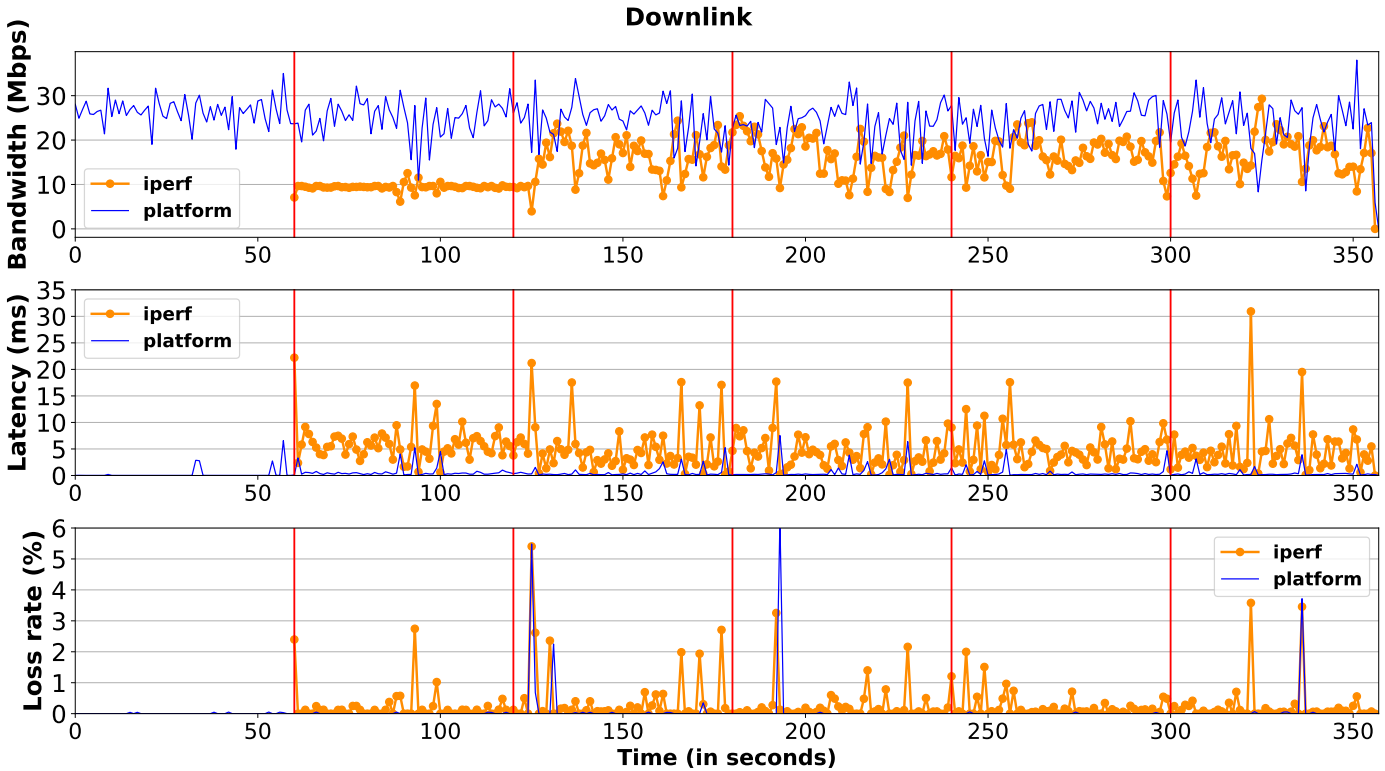
- QoS of CG traffic is preserved, on avg: 28.5Mb/s, 1ms latency, 0.2% loss
- TCP has high queuing delays (100ms on avg) and a reasonable bandwidth share (19.6 Mb/s).

HTB: SCReAM vs BBRv2



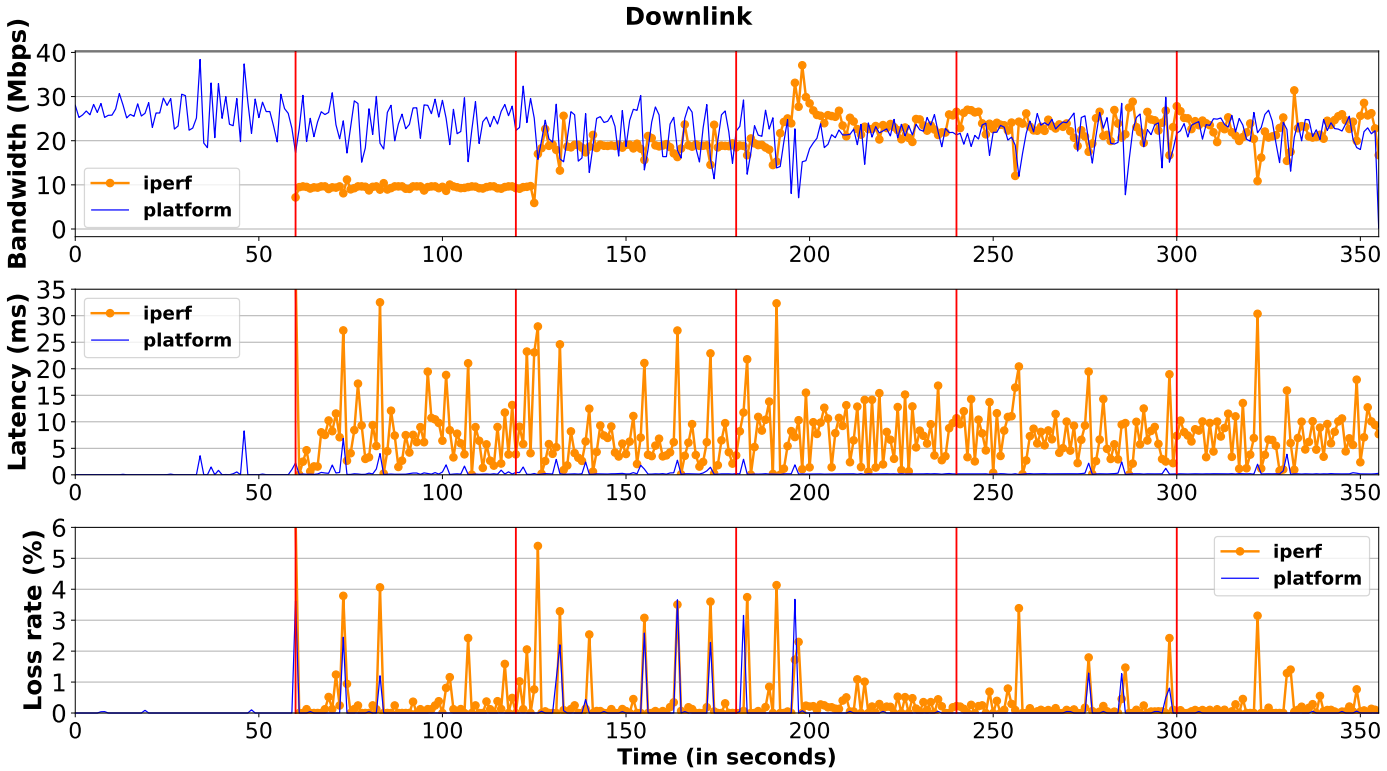
- QoS of CG traffic also perfectly preserved
- BBRv2 has lower delays than CUBIC (67ms on avg) and similar bitrate

L4S: SCReAM vs Cubic



- Very good CG traffic QoS: 24.25 Mbps, 0.5ms, <0.1% loss
- TCP traffic latency much reduced (4.7ms on avg vs 100ms with HTB), but lower throughput (16.43 vs 19.56 Mbps)
- Lower link utilization: 81.36% vs 96.4% with HTB

L4S: SCReAM vs BBRv2



- Very good CG traffic QoS: 21.66 Mbps, 0.3ms, 0.1% loss.
- Perfectly balanced bandwidth sharing around 22mbps.
- Still disappointing link utilization 86.7% vs 95.9% with HTB

Synthetic table

		Droptail		HTB		L4S	
		Cubic	BBRv2	Cubic	BBRv2	Cubic	BBRv2
CG Traffic	Avg Latency	44.9 ms	26.5 ms	1.16 ms	0.61 ms	0.53 ms	0.31 ms
	Variance	24.34	11.93	0.92	0.8	0.99	0.51
	% > 10 ms	96.28%	84.23%	0%	0%	0%	0%
	% > 20 ms	85.54%	68.88%	0%	0%	0%	0%
	% > 50 ms	34.71%	0.41%	0%	0%	0%	0%
	% > 100 ms	0%	0%	0%	0%	0%	0%
	Avg Bitrate	12.83 Mbps	15.18 Mbps	28.64 Mbps	28.66 Mbps	24.25 Mbps	21.66 Mbps
	1st quartile	4.29 Mbps	9.69 Mbps	27.22 Mbps	27.63 Mbps	21.83 Mbps	19.78 Mbps
	2nd quartile	12.17 Mbps	12.6 Mbps	29.14 Mbps	28.9 Mbps	25.06 Mbps	22.32 Mbps
	3rd quartile	18.81 Mbps	18.98 Mbps	30.78 Mbps	30.23 Mbps	27.52 Mbps	24.34 Mbps
	Loss Rate	0.25%	0.33%	0.24%	0.05%	0.08%	0.1%
TCP Traffic	Avg Latency	44.45 ms	25.96 ms	99.78 ms	67.38 ms	4.68 ms	7.57 ms
	Variance	24.57	12.31	26.84	20.91	4.03	5.67
	% > 10 ms	99.18%	84.71%	99.18%	99.17%	6.75%	25.42%
	% > 20 ms	82.3%	66.53%	99.18%	98.76%	0.84%	4.66%
	% > 50 ms	35.8%	0.83%	96.3%	81.41%	0%	0%
	% > 100 ms	0%	0%	48.15%	2.48%	0%	0%
	Avg Bitrate	33.34 Mbps	31.66 Mbps	19.56 Mbps	19.28 Mbps	16.43 Mbps	21.68 Mbps
	1st quartile	21.14 Mbps	21.73 Mbps	18.12 Mbps	17.55 Mbps	13.52 Mbps	19 Mbps
	2nd quartile	31.69 Mbps	32.74 Mbps	19.54 Mbps	19.31 Mbps	16.56 Mbps	22.25 Mbps
	3rd quartile	43.91 Mbps	38.31 Mbps	21.32 Mbps	20.73 Mbps	19.58 Mbps	24.08 Mbps
	Loss Rate	0.42%	0.57%	0.56%	0.16%	0.25%	0.33%
Link utilization	CG Traffic	25.65%	30.35%	57.27%	57.32%	48.5%	43.33%
	TCP Traffic	66.68%	63.31%	39.12%	38.57%	32.86%	43.37%
	Total	92.33%	93.66%	96.39%	95.89%	81.36%	86.7%

HTB operational advantages and constraints

HTB advantages

- Perfect isolation between traffic classes
- No need for application support

HTB limitations

- Needs knowledge of traffic to configure each class parameters: Ceil (maximum outgoing bandwidth capacity), Rate (guaranteed bandwidth), Prio (class priority)
- ML models needs to be trained regularly
- Classification errors can lead to net-neutrality violation

L4S operational advantages and constraints

L4S advantages

- No need for traffic classification
- Less configuration from ISP: only the coupling factor k (determines the balance between L4S and Classic flow rates)

L4S limitations

- Requires application support (CCA compliant with L4S)
- DualPi2 still couples queues: unresponsive traffic can steal bandwidth (ex:BBRv1)
- DualPi2 congestion marking probabilities defined after DCTCP / TCP CUBIC, less adapted to our case

Conclusion

Takeaway

- New open source experimental CG platform featuring Ericsson's SCReAM CCA
- CG traffic suffers from bufferbloat, competition with other CCA is unfair
- Both HTB / L4S queuing solutions preserve CG traffic QoS (1st time that L4S is evaluated with a real LL use-case)
- HTB / L4S have very different operational constraints

Future work

- Evaluate HTB and L4S for CG in cellular network conditions
- Consider more representative competing traffic (DASH video)
- Evaluate competing traffic in the low-latency queue

Thank you for your attention.

Any questions?